Posted 25 January 1998

Forth Philosophy 1998

Glen B. Haydon, M.D.
Epsilon Lyra Corporation
Route 2 Box 429
La Honda, CA 94020-9726

> I would like to build my Forth Philosophy this year around the concept that there is strength in diversity. Over the years, Forth has evolved in many divers ways. Individuals have taken different directions and the vendors have gone their own ways. Trying to force all Forth into a single mold will distract from its potential.

Forth as an Operating System

First off, Forth was designed as an operating system. As such it improved the efficiency of the application. However, it restricted the system on which it could be used.

Even with that limitation, Chuck brought up most of his implementations on dedicated systems and directed them to specific applications including any necessary operating system.

It is interesting that again there is discussion of making Forth the operating system after having taken a turn as only one program available in other operating systems. Perhaps this idea needs closer examination.

Early Public Forth Implementations

Early on, The Forth Primer from the Kitt Peak National Observatory provided a tutorial to the implementations on a set of minicomputers.

But it was not until the fig Forth implementation was made available as the Forth Installation Manual, that the language began to spread.

Several groups were taken with the power of the language and the Forth Interest Group eventually published source code for the implementation of Forth on some 12 different platforms.

At this time the interest in Forth exploded. The monthly fig Forth meetings were well attended. Regularly 150 to 200 were present at these meetings. There were problems and individuals contributed to solutions. The openness of the source and the solutions was stimulating.

Vendor Forth Implementations

At about this time several vendors developed and marketed their Forth implementations. There were numerous discussions about the need for more implementations of the Forth kernels. Each company had a good one. It was time to forget about implementing Forth and get on with the work of Forth applications.

It is interesting that most of the Forth Vendors could not make a business on sale of their implementations alone. As far as I know they all provided custom services for specific applications. Vendors often tailor their kernels for each application.

The early conflicts in goals among Forth users stimulated a wide variety of directions. The differences were always well discussed at fig meetings and conferences. The discussions were lively and comments were at times a little on the sharp side.

The Forth Heyday

This was the time of the heyday of Forth interest. We had a Byte issue which featured Forth. We had nearly 5000 members of fig. The FORML meetings were attended by 100 or so interested Forth users. For a few years we had an annual Forth Day which ran on to two days and were attended by 1000-2000 interested people.

A Change

This review of history is designed to highlight some of the diverse events which led to a rapid growth of interest in Forth. There was something about the philosophy at the time which was positive and stimulating.

Since then, the size of the Forth Interest Group has gradually decreased. We hear less and less about Forth in the general literature. The Rochester Conferences have ended after 18 years.

It is not clear that these FORML Conferences will continue after 20 years. What happened? The Standards Team entered the picture. First with the 79 Standard, then the 83 Standard, and then over a period of years the ANS Standard was developed. At the same time, fig offered no implementations for the new Standards as they came out. This was left to the Vendors.

I see the problem to be that Forth, in contrast with other languages, was originated as a language closely related to the application and the system being used. Most programmers are not familiar with closely bound hardware and software.

A second problem was that most users wanted to use their computers for other programs too. These systems came with an operating system to manage the other programs.

Forth became just another program. True, by renaming the Forth program COMMAND.COM, the Forth program would become the operating system.

However, it did not allow ready access to other programs on the same system.

Then there was demand for Forth Libraries and other tools to which programmers had become accustomed. Programmers new to Forth thought it had to look and work like other languages.

Hardware Evolution

Over the years, the computer systems have become more and more complex. Memory managers have been added under the control of the existing operating system. The disk managers were needed to allocate space on the floppy and then the hard disks in ever increasing size and used for many different programs.

Forth, originating as an operating system closely linking the hardware to the program for optimal efficiency, lost out for many of the major systems on the market.

Windows Operating Systems

Next came the windows paradigm starting with the Star system at Xerox Palo Alto Park. Over years, programmers became accustomed to a windows presentation. The operating system exploded in size and complexity.

Forth became a smaller and smaller part of the common computer systems. At present, many people are working to find a niche for Forth in an ever larger and more complex world.

A Return to Diversity

In a review of the history of Forth, we see that it has been used in diverse ways. A major trend has been away from hardware and implementations. Other features have been added such as libraries. The problem with libraries is that they require a standard kernel. The advantage is that they provide the tools programmers have come to expect.

This is the direction that Forth seems to be taking as far as development is concerned. But there are many other uses of Forth being actively implemented in embedded processors. We need to be aware of these many embedded application where Forth can be a major contender for programmers.

A Development Platform

A Development Platform includes many tools. Not all of them are easily implemented in Forth. One of the major tools which most Forth Programmers ignore completely is version control. As applications get large with many programmers and changing requirements, it is necessary to maintain version control for maintaining programs in the future.

Then there is the problem of drivers for the many additions to hardware. Each controller seems to do things just a little differently. Separate modules for each

new device can be used if the Development Platform allows linking in the various components.

In such an environment, Forth becomes yet another tool. In many cases Forth is an ideal tool. This seems to be a direction that some of Forth is moving.

Embedded Systems

There are a number of simple hardware systems under computer control. These systems are dedicated to the application. There is no need for all of the over head of a full Development Platform. The Forth kernel can be very small and the dialect used is of little importance. The program need not be portable.

With such Systems we are almost back to the origins of Forth.

There is a surprising number of such applications. A great diversity of Forth implementations continues to be developed.

Embedded Program Development

In many ways, it is convenient to develop the application on the system on which it is going to run. Often a simple I/O arrangement is sufficient. There is no need for anything more complicated.

On the other hand, a fully equipped Development Platform can be convenient. A simulator of the embedded system can be written. The application can be target compiled for the embedded process. The tools of program control can be invoked. The embedded system can be tethered to the Development Platform and tested.

It is a choice for the programmer of the embedded system. How much time does he want to spend to learn the power of a fully endowed Development Platform, and how much time is necessary to develop the embedded application.

The answer is in diversity. We need to get beyond the Forth "choir" to the many diverse users and encourage all of their diverse ways.

A Model Operating System

Linux is an open operating system with many participants world wide. It can be very large and is certainly complex.

Some months ago, I attended a Linux Day. Actually, it was only an evening. It was attended by about 1000 people. It reminded me of the Forth Days of years past. The enthusiasm of the participants was amazing. They must be doing something right.

In thinking about what is going on, I can see many parallels to what we went through with Forth. I can see where they may well end up with some of the same problems we have in Forth.

However, for those in need of a fully equipped Development System, Linux might

just provide the framework. Forth could provide another tool on a Linux platform. But, a minimal Forth as the operating system and application may be all that is necessary. I think that Forth users are scattered over the entire spectrum.

A Forth Philosophy of Diversity

It is time to open Forth to all. There is no requirement that any program is necessarily standard.

Conclusion

As Steven Pepper said in World Hypothesis, "I am dogmatically undogmatic."

Updated 21 August 1998

LEVELS OF FORTH

Glen B. Haydon
Mountain View Press
Box 429 Route 2
La Honda, CA 94020

This paper was presented at the 1991 Rochester Forth Conference.

Abstract

Match the level of Forth with the user and the application.

Introduction

Ontogeny recapitulate phylogeny. I guess it was John Dewey who emphasized that you start a student where he is. The needs of novices perhaps evolve in somewhat the same way as the history. If you are already at the end, you have no place to go.

A problem people in the Forth community face is the variety of applications which lend themselves to the language and the diversity of experience of the potential Forth users. One way to address the problems is to identify progressive levels of Forth with which to match the users and the applications. These Levels follow the history.

Level 0.

On several occasions C.H. Moore has listed approximately 63 functions which he considers the essence of Forth. I noted them at one presentation as follows:

```
+ - * /MOD MIN MAX = AND OR XOR
```

```
NEGATE ABS NOT */ DUP DROP SWAP OVER

DECIMAL HEX OCTAL . n .R

CR EMIT KEY : ; CREATE , ALLOT

IF ELSE THEN FOR NEXT I
```

There are only 45 functions in this list. You can see about which period this was. The list does not include `BEGIN`, `UNTIL`, and several other functions belong in Level 0. A figure of about 63 is appropriate.

Three functions are used for output to the display. He included no functions for any kind of storage. Screens are not included much less any means of writing on such a screen.

Something more than Level 0 is necessary to provide anyone but Chuck enough to work with.

Level 1.

The fig-Forth Installation Manual provides a complete set of Forth functions. The implementation with the Installation Manual is for the 6502. Since then, over a dozen implementations of fig-Forth for other processors were written and are still available in hard copy. About forty hours of careful typing will allow one to enter a listing.

At that time we lacked an editor. It was a chicken and egg problem. Though one had screens for compiling code there was no way to write to a screen until an editor was included. A simple single function editor allowed one to place code on a specific line of an already listed screen.

The fig-Forth model served as a wonderful learning tool. There was little in the way of documentation beyond the Installation Manual and the source listings. A few computer teaching programs did appear. The older publications such as the Kitt Peak Primer and Using Forth did not mesh well with the public domain version available.

Execellent applications have been done in fig-Forth.

The fig-Forth implementations took over the disk precluding its use for other programs. The error messages were stored on a disk.

Soon there came the 79 Standard which made several changes. It was then proposed that the Forth Interest Group stop supplying the fig-Forth source listings.

Level 2.

The 79 Standard made about 40 changes to the functions in fig-Forth. These made some things much easier. About the same time Leo Brody came out with the first edition of Starting Forth. This is an excellent book for a beginner. He developed the first part of the book in a very careful and artistic manner. Then he had to get on with the job.

At that time, Mountain View Press came out with a Forth implementation incorporating the 79 Standard changes. It put the error messages in line and added supplemental functions so that all but one of the functions matched those covered in Starting Forth. The single problem was 'tic'.

The assembler for several processors was included. A Forth editor described in Chapter 3 of Starting Forth written by Sam Daniels is included. Note: Starting Forth does not point out that you have to invoke the `EDITOR` vocabulary for it to work!

The complete source listing of MVP Forth was published for three processors. The appropriate source is included on each distribution disk. The Glossary, *ALL ABOUT FORTH* 2nd Edition, included the functional definitions, source code for the 8080, examples and comments for each of about 240 functions. The 3rd. Edition of *ALL ABOUT FORTH* has been expanded to include some 500 functions in common usage.

The distribution Version 03 of MVP Forth has remained unchanged since April 1983. Unfortunately several copies which have appeared on bulletin boards while being designated MVP Forth have been modified. The original Version 3 is now available of the Gene Forth Round Table and the Mountain View Press bulletin board.

The MVP Forth implementation takes over a disk format. But the file on Gene includes several added utilities which allow one to access screens in files and the compiling of text files written with any word processor.

Most of the Forth text books and tutorials begin with a discussion of stacks. After all it is the stacks which makes Forth what it is. But for most beginners stacks are a complication and a distraction. In the *FORTH GUIDE*, a different introduction is used.

The hardware designer and engineer have other interests besides learning yet another language. A small language which allows him to get on with his work would be sufficient. Levels 1 and 2 are closely bound to the hardware and serves such programmers well.

Level 3.

Several developments led to a new implementation. The 83 Standard was finally adopted. Few people had a computer dedicated to Forth. They needed their

computer for other things. There was also a growing library of Forth functions which could be incorporated.

F83, by Henry Laxon and Mike Perry, took the next step. F83 includes nearly 1000 functions. A substantial library of Forth functions was added. Ting's book, *Inside F83*, provided much documentation.

The most popular implementation of F83 is for the Intel based systems. It makes use of screens but now in files. A dedicated disk is no longer necessary and programs could be developed and run from a hard disk. Many of the functions were optimized assembler code which improved the speed. With the library of functions, we were on the way to an implementation of Forth for programmers. Ting's book, *Inside F83*, was a big help in printed documentation.

But F83 with its increased vocabulary presented a greater learning hurdle for the engineer and hardware type who had no interest in programming. There was much more to learn than was necessary for him. Of course, if he learned the language, he had more power available to him.

Level 4.

With a programmers demand for a more complete development system Tom Wempe developed his PADS system. At the same time, Tom Zimmer with many collaborators, went through many verisions of what became known as FPC>.

FPC is a full Forth Development System with a built in editor, debugger and many other features making it an outstanding package. However, the vocabulary has grown to nearly 2500 functions - A real learning challenge.

All of the source code is in text files. In fact, there are more source and help files in FPC than there are functions in fig-Forth. But the many features do not detract from speed for the experienced user. All possible speed is milked out of the processor. When added to the TCOM program small target compiled applications can be made.

The program and files are large. One needs a couple of megabytes of disk space available. To do all that it does, the space is needed. For the professional programmer, FPC is an excellent model. Ting has provided us the *F-PC Technical Reference Manual* and the *F-PC Users Manual.*

But for the engineer and hardware designer who is not interested in programming languages, the learning curve has increased. It may be more than he wants. However, with such a development system, Tom Zimmer has shown that hardware problems can be easily solved.

Level 5.

For many purposes, 32-bit values are overkill, but they are the coming processors. The segmented architecture of the Intel processors has led to severe limitations.

Various schemes of memory management have been developed to work around some of those limitations. A 32-bit Forth system should have 32-bit stacks and address space. Some means should be found to deal efficiently with 16-bit addresses in the segmented architecture.

F32, by Rick VanNorman, solves some of these issues. He has created a full development package including a text editor and assembler in about 900 words. This has been a outgrowth of his years of experience with Forth. He has placed the program in the public domain and is available from several sources including the Mountain View Press bulletin board.

His code is neat and well organized providing reasonable documentation for an experienced Forth programmer. However, there is as yet no printed documentation available. I expect that there will be some before long. This implementation is not intended for a beginning Forth programmer.

Discussion.

These Levels are oriented toward systems based on Intel processors. Most of them operate under a PC-DOS or MS-DOS environment. Other systems also make use of Forth. The Sun work station is built with a Forth monitor, and runs Unix. The Macintosh has available several Forth implementations. A Forth like Yerkes evolved out of NEON and runs on the Macintosh. These implementations are not for the beginner who wants a small language to get started. World wide there are still more implementations of Forth.

This is not to ignore the many professional Forth implementations. I consider these levels of Forth as models. Depending on ones interest and time, he can delve as deeply as he wishes.

For professional programmers, the professionally written implementations are well honed. They provide full development platforms. The big advantage of the professional systems is the many extensions and the level of documentation and support available from the respective companies.

The professional products include an appropriate tutorial. I have read the documentation and tried a number of the professional implementations. They are good. Having an interest in implementation as well, I find the hidden proprietary nature of most of them to be frustrating. For a programmer these works offer the professional touch not found in the public domain implementations.

The problem then boils down to determining the programming level of the user and his applications. Select a tutorial or instructional manual and work through it. The order in each is different. Trying to learn from several at once will only confuse the student. Don't demand that the hardware designer and engineer be master computer scientists as well. Don't make the computer scientist work with minimal amateurish tools.

I think that there is a niche for Forth. The full blown development systems,

whatever the language used, is overwhelming for many engineer and hardware designers. The interactive capability of Forth supplies these users a rapid means of progress. A small set of functions which meet his needs is sufficient.

All users are not equal.

Addendum:

Added and Modified 24 March 1995

Since this article was written, in 1994 an ANSI Forth Standard has been adopted. This dialect is the result of many man years of work by experienced Forth programmers. A number of people have attempted to implement this dialect and in 32 bit address space for the PC and clones.

Unfortunately, the ANSI Standard Document is priced at nearly $300.0 (sic) and will be hard for may Forth users to justify. However, a hypertext copy of the final draft before adoption is available.

Also there is an implementation, THIS4TH, which is available in C source code and complied to a number of systems. These are also available.

Another implementation which is near the ANSI Standard Forth for use on PC's in 32 bit address space using DPMI. DPMI is available when shelling out from WINDOWS. This was written by Rick van Norman and is contained in S4.ZIP.

There will be other public domain implementations and these will all be tested by use. Eventually, there will probably appear selected versions and some documentation will be written.

The goal of these Forth Levels and these comments is to provide the beginner with something of a perspective.

Glen B. Haydon, M.D.
ghaydon@theforthsource.com

Updated 21 August 1998

LEVELS OF FORTH

Glen B. Haydon
Mountain View Press
Box 429 Route 2
La Honda, CA 94020

This paper was presented at the 1991 Rochester Forth Conference.

Abstract

Match the level of Forth with the user and the application.

Introduction

Ontogeny recapitulate phylogeny. I guess it was John Dewey who emphasized that you start a student where he is. The needs of novices perhaps evolve in somewhat the same way as the history. If you are already at the end, you have no place to go.

A problem people in the Forth community face is the variety of applications which lend themselves to the language and the diversity of experience of the potential Forth users. One way to address the problems is to identify progressive levels of Forth with which to match the users and the applications. These Levels follow the history.

Level 0.

On several occasions C.H. Moore has listed approximately 63 functions which he considers the essence of Forth. I noted them at one presentation as follows:

```
+ - * /MOD MIN MAX = AND OR XOR

NEGATE ABS NOT */ DUP DROP SWAP OVER

DECIMAL HEX OCTAL . n .R

CR EMIT KEY : ; CREATE , ALLOT

IF ELSE THEN FOR NEXT I
```

There are only 45 functions in this list. You can see about which period this was. The list does not include `BEGIN`, `UNTIL`, and several other functions belong in Level 0. A figure of about 63 is appropriate.

Three functions are used for output to the display. He included no functions for any kind of storage. Screens are not included much less any means of writing on such a screen.

Something more than Level 0 is necessary to provide anyone but Chuck enough to work with.

Level 1.

The fig-Forth Installation Manual provides a complete set of Forth functions. The

implementation with the Installation Manual is for the 6502. Since then, over a dozen implementations of fig-Forth for other processors were written and are still available in hard copy. About forty hours of careful typing will allow one to enter a listing.

At that time we lacked an editor. It was a chicken and egg problem. Though one had screens for compiling code there was no way to write to a screen until an editor was included. A simple single function editor allowed one to place code on a specific line of an already listed screen.

The fig-Forth model served as a wonderful learning tool. There was little in the way of documentation beyond the Installation Manual and the source listings. A few computer teaching programs did appear. The older publications such as the Kitt Peak Primer and Using Forth did not mesh well with the public domain version available.

Execellent applications have been done in fig-Forth.

The fig-Forth implementations took over the disk precluding its use for other programs. The error messages were stored on a disk.

Soon there came the 79 Standard which made several changes. It was then proposed that the Forth Interest Group stop supplying the fig-Forth source listings.

Level 2.

The 79 Standard made about 40 changes to the functions in fig-Forth. These made some things much easier. About the same time Leo Brody came out with the first edition of Starting Forth. This is an excellent book for a beginner. He developed the first part of the book in a very careful and artistic manner. Then he had to get on with the job.

At that time, Mountain View Press came out with a Forth implementation incorporating the 79 Standard changes. It put the error messages in line and added supplemental functions so that all but one of the functions matched those covered in Starting Forth. The single problem was 'tic'.

The assembler for several processors was included. A Forth editor described in Chapter 3 of Starting Forth written by Sam Daniels is included. Note: Starting Forth does not point out that you have to invoke the `EDITOR` vocabulary for it to work!

The complete source listing of MVP Forth was published for three processors. The appropriate source is included on each distribution disk. The Glossary, *ALL ABOUT FORTH* 2nd Edition, included the functional definitions, source code for the 8080, examples and comments for each of about 240 functions. The 3rd. Edition of *ALL ABOUT FORTH* has been expanded to include some 500 functions in common usage.

The distribution Version 03 of MVP Forth has remained unchanged since April 1983. Unfortunately several copies which have appeared on bulletin boards while being designated MVP Forth have been modified. The original Version 3 is now available of the Gene Forth Round Table and the Mountain View Press bulletin board.

The MVP Forth implementation takes over a disk format. But the file on Gene includes several added utilities which allow one to access screens in files and the compiling of text files written with any word processor.

Most of the Forth text books and tutorials begin with a discussion of stacks. After all it is the stacks which makes Forth what it is. But for most beginners stacks are a complication and a distraction. In the *FORTH GUIDE*, a different introduction is used.

The hardware designer and engineer have other interests besides learning yet another language. A small language which allows him to get on with his work would be sufficient. Levels 1 and 2 are closely bound to the hardware and serves such programmers well.

Level 3.

Several developments led to a new implementation. The 83 Standard was finally adopted. Few people had a computer dedicated to Forth. They needed their computer for other things. There was also a growing library of Forth functions which could be incorporated.

F83, by Henry Laxon and Mike Perry, took the next step. F83 includes nearly 1000 functions. A substantial library of Forth functions was added. Ting's book, *Inside F83*, provided much documentation.

The most popular implementation of F83 is for the Intel based systems. It makes use of screens but now in files. A dedicated disk is no longer necessary and programs could be developed and run from a hard disk. Many of the functions were optimized assembler code which improved the speed. With the library of functions, we were on the way to an implementation of Forth for programmers. Ting's book, *Inside F83*, was a big help in printed documentation.

But F83 with its increased vocabulary presented a greater learning hurdle for the engineer and hardware type who had no interest in programming. There was much more to learn than was necessary for him. Of course, if he learned the language, he had more power available to him.

Level 4.

With a programmers demand for a more complete development system Tom Wempe developed his PADS system. At the same time, Tom Zimmer with many collaborators, went through many verisions of what became known as FPC>.

FPC is a full Forth Development System with a built in editor, debugger and many other features making it an outstanding package. However, the vocabulary has grown to nearly 2500 functions - A real learning challenge.

All of the source code is in text files. In fact, there are more source and help files in FPC than there are functions in fig-Forth. But the many features do not detract from speed for the experienced user. All possible speed is milked out of the processor. When added to the TCOM program small target compiled applications can be made.

The program and files are large. One needs a couple of megabytes of disk space available. To do all that it does, the space is needed. For the professional programmer, FPC is an excellent model. Ting has provided us the *F-PC Technical Reference Manual* and the *F-PC Users Manual.*

But for the engineer and hardware designer who is not interested in programming languages, the learning curve has increased. It may be more than he wants. However, with such a development system, Tom Zimmer has shown that hardware problems can be easily solved.

Level 5.

For many purposes, 32-bit values are overkill, but they are the coming processors. The segmented architecture of the Intel processors has led to severe limitations. Various schemes of memory management have been developed to work around some of those limitations. A 32-bit Forth system should have 32-bit stacks and address space. Some means should be found to deal efficiently with 16-bit addresses in the segmented architecture.

F32, by Rick VanNorman, solves some of these issues. He has created a full development package including a text editor and assembler in about 900 words. This has been a outgrowth of his years of experience with Forth. He has placed the program in the public domain and is available from several sources including the Mountain View Press bulletin board.

His code is neat and well organized providing reasonable documentation for an experienced Forth programmer. However, there is as yet no printed documentation available. I expect that there will be some before long. This implementation is not intended for a beginning Forth programmer.

Discussion.

These Levels are oriented toward systems based on Intel processors. Most of them operate under a PC-DOS or MS-DOS environment. Other systems also make use of Forth. The Sun work station is built with a Forth monitor, and runs Unix. The Macintosh has available several Forth implementations. A Forth like Yerkes evolved out of NEON and runs on the Macintosh. These implementations are not for the beginner who wants a small language to get started. World wide there are still more implementations of Forth.

This is not to ignore the many professional Forth implementations. I consider these levels of Forth as models. Depending on ones interest and time, he can delve as deeply as he wishes.

For professional programmers, the professionally written implementations are well honed. They provide full development platforms. The big advantage of the professional systems is the many extensions and the level of documentation and support available from the respective companies.

The professional products include an appropriate tutorial. I have read the documentation and tried a number of the professional implementations. They are good. Having an interest in implementation as well, I find the hidden proprietary nature of most of them to be frustrating. For a programmer these works offer the professional touch not found in the public domain implementations.

The problem then boils down to determining the programming level of the user and his applications. Select a tutorial or instructional manual and work through it. The order in each is different. Trying to learn from several at once will only confuse the student. Don't demand that the hardware designer and engineer be master computer scientists as well. Don't make the computer scientist work with minimal amateurish tools.

I think that there is a niche for Forth. The full blown development systems, whatever the language used, is overwhelming for many engineer and hardware designers. The interactive capability of Forth supplies these users a rapid means of progress. A small set of functions which meet his needs is sufficient.

All users are not equal.

Addendum:

Added and Modified 24 March 1995

Since this article was written, in 1994 an ANSI Forth Standard has been adopted. This dialect is the result of many man years of work by experienced Forth programmers. A number of people have attempted to implement this dialect and in 32 bit address space for the PC and clones.

Unfortunately, the ANSI Standard Document is priced at nearly $300.0 (sic) and will be hard for may Forth users to justify. However, a hypertext copy of the final draft before adoption is available.

Also there is an implementation, THIS4TH, which is available in C source code and complied to a number of systems. These are also available.

Another implementation which is near the ANSI Standard Forth for use on PC's in 32 bit address space using DPMI. DPMI is available when shelling out from WINDOWS. This was written by Rick van Norman and is contained in S4.ZIP.

There will be other public domain implementations and these will all be tested by use. Eventually, there will probably appear selected versions and some documentation will be written.

The goal of these Forth Levels and these comments is to provide the beginner with something of a perspective.

Glen B. Haydon, M.D.
ghaydon@theforthsource.com